Министерство науки и образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Волгоградский государственный технический университет»

Утверждаю

Зав. кафедрой ПОАС, проф. д.т.н.

\_\_\_\_\_\_\_\_\_\_\_\_\_ Дворянкин А. М.

«\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_ 2017 г.

КАЧЕСТВО И НАДЕЖНОСТЬ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Определить интерфейс функции при ее выделении из кода программы

Техническое задание

|  |  |
| --- | --- |
| СОГЛАСОВАНО:Руководитель проекта:Доцент кафедры ПОАС\_\_\_\_\_\_\_\_\_\_\_\_\_ Сычев О. А.«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2017 г. | Разработчик:Студент ПрИн-266\_\_\_\_\_\_\_\_\_\_\_\_ Катышев А. М.«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2017 г. |
|  | Нормоконтролер:Ассистент кафедры ПОАС\_\_\_\_\_\_\_\_\_\_\_\_\_ Мамонтов Д. П.«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2017 г. |

Волгоград, 2017 г.

Содержание

[Введение 3](#_Toc476509963)

[1 Основание для разработки 3](#_Toc476509964)

[2 Назначение разработки 3](#_Toc476509965)

[3 Требования к программе 4](#_Toc476509966)

[3.1 Требования к функциональным характеристикам 4](#_Toc476509967)

[3.2 Требования к надежности 5](#_Toc476509968)

[3.3 Требования к составу и параметрам технических средств 7](#_Toc476509969)

[3.4 Требования к информационной и программной совместимости 7](#_Toc476509970)

[4 Требования к программной документации 8](#_Toc476509971)

[5 Стадии и этапы разработки 8](#_Toc476509972)

[6 Порядок контроля и приемки 9](#_Toc476509973)

[Приложение А. Текстовый файл, содержащий границы участка кода программы и имя новой выделяемой функции 10](#_Toc476509974)

[Приложение Б. XML-файл, содержащий дерево разбора входной программы 11](#_Toc476509975)

[Приложение В. XML-файл, содержащий примерное дерево разбора выходной программы после проведения рефакторинга 15](#_Toc476509976)

[Приложение Г. Примеры ошибочных данных и сообщений, выдаваемых программой 21](#_Toc476509977)

[Приложение Д. Тэги и атрибуты XML-файла выделенной функции 31](#_Toc476509978)

# Введение

Название программы: Function Refactor.

Программа Function Refactor предназначена для проведения рефакторинга «Выделение функции» данного участка кода внутри исходного кода программы. Программа должна выделять логически завершенный участок кода, расположенный внутри известных границ исходного кода программы, в отдельную функцию, при использовании которой вместо выделенного фрагмента не изменялось бы внешнее поведение всей программы.

# 1 Основание для разработки

Разработка программы осуществляется на основании задания на курсовой проект по дисциплине «Качество и надежность программного обеспечения», выданного доцентом кафедры ПОАС Сычевым О.А. 8 февраля 2017 г.

# 2 Назначение разработки

Function Refactor может применяться для изменения внутренней структуры программ путем выделения определенного участка кода в отдельную функцию, не нарушая при этом их работоспособности.

# 3 Требования к программе

# 3.1 Требования к функциональным характеристикам

Программа получает в качестве входного параметрадерево разбора текста программы (абстрактное синтаксическое дерево) с объявлением всех переменных, а также границы внутри кода программы.

Требования к используемым данным в коде программы:

1. допускаются следующие базовые типы данных, используемых в программе: bool, int, float, double, char. Также могут использоваться пользовательский тип данных struct; поля внутри структуры могут иметь только типы, указанные выше (bool, int, float, double, char);
2. допускается квалификатор типов const;
3. допускается использование в коде программы одномерных массивов (многомерные массивы недопустимы), максимально возможный размер массива – 100 (размер массивов в тексте программы задается непосредственно: например – int a[5], массивы переменных размеров недопустимы);
4. дерево разбора текста программы не содержит в себе оператора sizeof при работе с массивами;
5. дерево разбора текста программы не содержит в себе оператора множественного выбора switch;
6. не допускается использование макросов внутри кода программы;
7. не допускается использование шаблонов функций, классов, методов и т.д. (template-производные);
8. исходный код программы «написан на процедурном языке», т.е. не допускается использование классов в коде программы (использование объединений и перечислений не допускается);
9. не допускается наличие в коде программы using-директив (но разрешены переопределения типов typedef).

В качестве выходного параметра выдается видоизмененное дерево разбора текста программы (абстрактное синтаксическое дерево) с проведенным рефакторингом «Выделение функции». К изначальному дереву добавляется новое дерево объявления функции с заданным именем, располагающееся перед кодом основной программы. Участок кода, находящийся внутри выделенных границ, заменяется на вызов выделенной функции.

# 3.2 Требования к надежности

Входные данные состоят из следующих файлов:

1. xml-файл, содержащий абстрактное синтаксическое дерево разбора программы на языке Си;
2. текстовый файл в формате txt, содержащий границы фрагмента кода в цифровом формате, подлежащего рефакторингу, а также имя новой выделяемой функции (формат и пример содержания подобного текстового файла см. приложение А).

Пример входной программы на языке Си, а также абстрактное синтаксическое дерево разбора входной программы см. приложение Б.

Состав выходных данных:

XML-файл, содержащий абстрактное синтаксическое дерево разбора программы на языке Си после проведения рефакторинга «Выделение функции».

Пример выходной программы на языке Си после проведения рефакторнга см. приложение В:

В случае нарушения структуры дерева разбора программы, необходимо выдать сообщение об ошибке и корректно завершить работу.

Случаи ошибочных данных текстового файла (примеры к нижеперечисленным случаям и примеры сообщений об ошибках см. приложение Г):

1) в случае ошибочных границ, находящихся за пределами текста программы, необходимо выдать сообщение об ошибке, содержащее границу, которая находится вне текста программы и корректно завершить работу;

2) в случае неверного порядка следования границ в текстовом файле, необходимо поменять местами вводимые границы и продолжить обработку данных без завершения работы;

3) в случае некорректного составления формата текстового файла, необходимо выдать сообщение об ошибке, с указанием конкретного места в файле, где нарушен формат, и корректно завершить работу;

4) в случае некорректного имени для новой выделяемой функции, (имя должно состоять только из латинских букв и/или цифр, причем первый символ имени должен быть латинской буквой или символом подчеркивания; имя функции не должно совпадать с другими идентификаторами, используемых в тексте программы) необходимо выдать сообщение об ошибочном вводе имени функции и корректно завершить работу.

В случае если провести рефакторинг «Выделение функции» невозможно (любые разрывы границами корректного участка кода: разрывы логических синтаксических блоков кода, разрывы любых видов выражений или вызовов других функций внутри фрагмента кода), необходимо выдать сообщение, содержащего строку с разрывом синтаксической структуры и корректно завершить работу.

Все примеры к вышеперечисленным ошибкам описаны в приложении Г.

# 3.3 Требования к составу и параметрам технических средств

В состав технических средств должен входить персональный компьютер (стационарный компьютер или ноутбук), включающий в себя:

- процессор Intel Core 2 Duo 2 ГГц или лучше;

- оперативную память объемом 512 Мб или больше;

- жесткий диск объемом не менее 200 Мб;

- операционную систему Windows XP или выше;

- наличие клавиатуры, мыши.

# 3.4 Требования к информационной и программной совместимости

Консольная программа Function Refactor принимает на вход два файла, имена которых передаются программе как параметры в командной строке:

- файл, содержащий абстрактное синтаксическое дерево разбора программы, в формате XML;

- файл, содержащий границы выделяемого фрагмента из кода программы с именем новой выделяемой функции, в формате txt.

Пример вызова программы из командной строки:

FunctionRefactor.exe program.xml data.txt

Абстрактное синтаксическое дерево исходного текста программы на языке Си формируется в сторонней программе, выполняющей его парсинг и предоставляющий на выход файл дерева разбора исходного текста программы в формате XML, который используется на входе программы Function Refactor .

На выходе программа Function Refactor выдает видоизмененное абстрактное синтаксическое дерево с учетом удачно проведенного рефакторинга в формате XML (тэги, относящиеся к выделенной функции, могут отличаться от общей структуры дерева исходной программы, но структуру должны передать корректно, см. приложение Д).

Программа Function Refactor должна быть написана на языке C++ с использованием библиотеки Qt.

# 4 Требования к программной документации

Состав программной документации должен включать в себя:

- в бумажной форме:

- техническое задание;

- технический проект в виде описания программы;

- программа и методика испытаний;

- руководство программиста;

- в электронной форме:

- копии всех документов бумажной формы;

- рабочая документация (генерируется doxygen);

- текст программы.

Вся документация должна быть представлена в соответствии с ГОСТ 19.

# 5 Стадии и этапы разработки

- техническое задание — до 10 марта 2017 г.;

- технический проект — до 7 апреля 2017 г.;

- рабочий проект — до 10 мая 2017 г.;

- внедрение — до 24 мая 2017 г.

# 6 Порядок контроля и приемки

Программа сдается для проверки преподавателю не позднее 10.05.2017г.

При обнаружении в программе ошибок и недостатков, исполнитель устраняет их в недельный срок и предоставляет программу на повторную проверку.

# Приложение А

Текстовый файл, содержащий границы участка кода программы и имя новой выделяемой функции

Формат данных, содержащихся в текстовом файле:

[индекс строки верхней границы] [пробел] [начальная позиция границы в этой строке]

[перевод строки]

[индекс строки нижней границы] [пробел] [конечная позиция границы в этой строке]

[перевод строки]

[имя новой выделяемой функции]

Пример содержания текстового файла для программы из приложения Б:

5 0

9 0

function

# Приложение Б

XML-файл, содержащий дерево разбора входной программы

<program linestart="0" colstart="0" lineend="11" colend="1">

 <stmt\_list linestart="0" colstart="0" lineend="11" colend="1">

 <stmt\_or\_defined\_macro linestart="0" colstart="0" lineend="11" colend="1">

 <builtintype linestart="0" colstart="0" lineend="0" colend="2">

 <typename value="int" linestart="0" colstart="0" lineend="0" colend="2" />

 </builtintype>

 <function\_definition\_without\_type linestart="0" colstart="4" lineend="11" colend="1">

 <identifier value="main" linestart="0" colstart="4" lineend="0" colend="7" />

 <formal\_args\_list\_with\_or\_without\_const linestart="0" colstart="8" lineend="0" colend="9">

 <args\_list linestart="0" colstart="8" lineend="0" colend="9">

 <bracket value="(" linestart="0" colstart="8" lineend="0" colend="8" />

 <bracket value=")" linestart="0" colstart="9" lineend="0" colend="9" />

 </args\_list>

 </formal\_args\_list\_with\_or\_without\_const>

 <function\_body linestart="1" colstart="1" lineend="11" colend="1">

 <bracket value="{" linestart="1" colstart="1" lineend="1" colend="1" />

 <stmt\_list linestart="2" colstart="2" lineend="10" colend="10">

 <stmt\_list linestart="2" colstart="2" lineend="8" colend="2">

 <stmt\_list linestart="2" colstart="2" lineend="3" colend="11">

 <stmt\_list linestart="2" colstart="2" lineend="2" colend="11">

 <stmt\_or\_defined\_macro linestart="2" colstart="2" lineend="2" colend="11">

 <stmt linestart="2" colstart="2" lineend="2" colend="11">

 <variable\_declaration linestart="2" colstart="2" lineend="2" colend="10">

 <builtintype linestart="2" colstart="2" lineend="2" colend="4">

 <typename value="int" linestart="2" colstart="2" lineend="2" colend="4" />

 </builtintype>

 <assign linestart="2" colstart="6" lineend="2" colend="10">

 <identifier value="a" linestart="2" colstart="6" lineend="2" colend="6" />

 <operators value="=" linestart="2" colstart="8" lineend="2" colend="8" />

 <numeric value="1" linestart="2" colstart="10" lineend="2" colend="10" />

 </assign>

 </variable\_declaration>

 <semicolon value=";" linestart="2" colstart="11" lineend="2" colend="11" />

 </stmt>

 </stmt\_or\_defined\_macro>

 </stmt\_list>

 <stmt\_or\_defined\_macro linestart="3" colstart="2" lineend="3" colend="11">

 <stmt linestart="3" colstart="2" lineend="3" colend="11">

 <variable\_declaration linestart="3" colstart="2" lineend="3" colend="10">

 <builtintype linestart="3" colstart="2" lineend="3" colend="4">

 <typename value="int" linestart="3" colstart="2" lineend="3" colend="4" />

 </builtintype>

 <assign linestart="3" colstart="6" lineend="3" colend="10">

 <identifier value="b" linestart="3" colstart="6" lineend="3" colend="6" />

 <operators value="=" linestart="3" colstart="8" lineend="3" colend="8" />

 <numeric value="2" linestart="3" colstart="10" lineend="3" colend="10" />

 </assign>

 </variable\_declaration>

 <semicolon value=";" linestart="3" colstart="11" lineend="3" colend="11" />

 </stmt>

 </stmt\_or\_defined\_macro>

 </stmt\_list>

 <stmt\_or\_defined\_macro linestart="5" colstart="2" lineend="8" colend="2">

 <stmt linestart="5" colstart="2" lineend="8" colend="2">

 <if\_then\_else linestart="5" colstart="2" lineend="8" colend="2">

 <if\_then linestart="5" colstart="2" lineend="8" colend="2">

 <keyword value="if" linestart="5" colstart="2" lineend="5" colend="3" />

 <bracket value="(" linestart="5" colstart="5" lineend="5" colend="5" />

 <expr\_greater linestart="5" colstart="6" lineend="5" colend="10">

 <identifier value="a" linestart="5" colstart="6" lineend="5" colend="6" />

 <operators value=">" linestart="5" colstart="8" lineend="5" colend="8" />

 <identifier value="b" linestart="5" colstart="10" lineend="5" colend="10" />

 </expr\_greater>

 <bracket value=")" linestart="5" colstart="11" lineend="5" colend="11" />

 <stmt linestart="6" colstart="2" lineend="8" colend="2">

 <bracket value="{" linestart="6" colstart="2" lineend="6" colend="2" />

 <stmt\_list linestart="7" colstart="3" lineend="7" colend="9">

 <stmt\_or\_defined\_macro linestart="7" colstart="3" lineend="7" colend="9">

 <stmt linestart="7" colstart="3" lineend="7" colend="9">

 <expr\_assign linestart="7" colstart="3" lineend="7" colend="8">

 <identifier value="a" linestart="7" colstart="3" lineend="7" colend="3" />

 <operators value="=" linestart="7" colstart="5" lineend="7" colend="5" />

 <numeric value="10" linestart="7" colstart="7" lineend="7" colend="8" />

 </expr\_assign>

 <semicolon value=";" linestart="7" colstart="9" lineend="7" colend="9" />

 </stmt>

 </stmt\_or\_defined\_macro>

 </stmt\_list>

 <bracket value="}" linestart="8" colstart="2" lineend="8" colend="2" />

 </stmt>

 </if\_then>

 </if\_then\_else>

 </stmt>

 </stmt\_or\_defined\_macro>

 </stmt\_list>

 <stmt\_or\_defined\_macro linestart="10" colstart="2" lineend="10" colend="10">

 <stmt linestart="10" colstart="2" lineend="10" colend="10">

 <keyword value="return" linestart="10" colstart="2" lineend="10" colend="7" />

 <numeric value="0" linestart="10" colstart="9" lineend="10" colend="9" />

 <semicolon value=";" linestart="10" colstart="10" lineend="10" colend="10" />

 </stmt>

 </stmt\_or\_defined\_macro>

 </stmt\_list>

 <bracket value="}" linestart="11" colstart="1" lineend="11" colend="1" />

 </function\_body>

 </function\_definition\_without\_type>

 </stmt\_or\_defined\_macro>

 </stmt\_list>

</program>

Представление данного дерева на языке Си:

int main()

{

int a = 1;

int b = 2;

if (a > b)

{

 a = 10;

}

return 0;

}

# Приложение В

XML-файл, содержащий примерное дерево разбора выходной программы после проведения рефакторинга

<program linestart="0" colstart="0" lineend="15" colend="1">

 <stmt\_list linestart="0" colstart="0" lineend="15" colend="1">

 <stmt\_list linestart="0" colstart="0" lineend="6" colend="1">

 <stmt\_or\_defined\_macro linestart="0" colstart="0" lineend="6" colend="1">

 <builtintype linestart="0" colstart="0" lineend="0" colend="3">

 <typename value="void" linestart="0" colstart="0" lineend="0" colend="3" />

 </builtintype>

 <function\_definition\_without\_type linestart="0" colstart="5" lineend="6" colend="1">

 <identifier value="function" linestart="0" colstart="5" lineend="0" colend="12" />

 <formal\_args\_list\_with\_or\_without\_const linestart="0" colstart="13" lineend="0" colend="26">

 <formal\_args\_list linestart="0" colstart="13" lineend="0" colend="26">

 <bracket value="(" linestart="0" colstart="13" lineend="0" colend="13" />

 <arg\_list linestart="0" colstart="14" lineend="0" colend="25">

 <arg\_list linestart="0" colstart="14" lineend="0" colend="18">

 <arg linestart="0" colstart="14" lineend="0" colend="18">

 <builtintype linestart="0" colstart="14" lineend="0" colend="16">

 <typename value="int" linestart="0" colstart="14" lineend="0" colend="16" />

 </builtintype>

 <identifier value="a" linestart="0" colstart="18" lineend="0" colend="18" />

 </arg>

 </arg\_list>

 <comma value="," linestart="0" colstart="19" lineend="0" colend="19" />

 <arg linestart="0" colstart="21" lineend="0" colend="25">

 <builtintype linestart="0" colstart="21" lineend="0" colend="23">

 <typename value="int" linestart="0" colstart="21" lineend="0" colend="23" />

 </builtintype>

 <identifier value="b" linestart="0" colstart="25" lineend="0" colend="25" />

 </arg>

 </arg\_list>

 <bracket value=")" linestart="0" colstart="26" lineend="0" colend="26" />

 </formal\_args\_list>

 </formal\_args\_list\_with\_or\_without\_const>

 <function\_body linestart="1" colstart="1" lineend="6" colend="1">

 <bracket value="{" linestart="1" colstart="1" lineend="1" colend="1" />

 <stmt\_list linestart="2" colstart="2" lineend="5" colend="2">

 <stmt\_or\_defined\_macro linestart="2" colstart="2" lineend="5" colend="2">

 <stmt linestart="2" colstart="2" lineend="5" colend="2">

 <if\_then\_else linestart="2" colstart="2" lineend="5" colend="2">

 <if\_then linestart="2" colstart="2" lineend="5" colend="2">

 <keyword value="if" linestart="2" colstart="2" lineend="2" colend="3" />

 <bracket value="(" linestart="2" colstart="5" lineend="2" colend="5" />

 <expr\_greater linestart="2" colstart="6" lineend="2" colend="10">

 <identifier value="a" linestart="2" colstart="6" lineend="2" colend="6" />

 <operators value=">" linestart="2" colstart="8" lineend="2" colend="8" />

 <identifier value="b" linestart="2" colstart="10" lineend="2" colend="10" />

 </expr\_greater>

 <bracket value=")" linestart="2" colstart="11" lineend="2" colend="11" />

 <stmt linestart="3" colstart="2" lineend="5" colend="2">

 <bracket value="{" linestart="3" colstart="2" lineend="3" colend="2" />

 <stmt\_list linestart="4" colstart="3" lineend="4" colend="9">

 <stmt\_or\_defined\_macro linestart="4" colstart="3" lineend="4" colend="9">

 <stmt linestart="4" colstart="3" lineend="4" colend="9">

 <expr\_assign linestart="4" colstart="3" lineend="4" colend="8">

 <identifier value="a" linestart="4" colstart="3" lineend="4" colend="3" />

 <operators value="=" linestart="4" colstart="5" lineend="4" colend="5" />

 <numeric value="10" linestart="4" colstart="7" lineend="4" colend="8" />

 </expr\_assign>

 <semicolon value=";" linestart="4" colstart="9" lineend="4" colend="9" />

 </stmt>

 </stmt\_or\_defined\_macro>

 </stmt\_list>

 <bracket value="}" linestart="5" colstart="2" lineend="5" colend="2" />

 </stmt>

 </if\_then>

 </if\_then\_else>

 </stmt>

 </stmt\_or\_defined\_macro>

 </stmt\_list>

 <bracket value="}" linestart="6" colstart="1" lineend="6" colend="1" />

 </function\_body>

 </function\_definition\_without\_type>

 </stmt\_or\_defined\_macro>

 </stmt\_list>

 <stmt\_or\_defined\_macro linestart="7" colstart="1" lineend="15" colend="1">

 <builtintype linestart="7" colstart="1" lineend="7" colend="3">

 <typename value="int" linestart="7" colstart="1" lineend="7" colend="3" />

 </builtintype>

 <function\_definition\_without\_type linestart="7" colstart="5" lineend="15" colend="1">

 <identifier value="main" linestart="7" colstart="5" lineend="7" colend="8" />

 <formal\_args\_list\_with\_or\_without\_const linestart="7" colstart="9" lineend="7" colend="10">

 <args\_list linestart="7" colstart="9" lineend="7" colend="10">

 <bracket value="(" linestart="7" colstart="9" lineend="7" colend="9" />

 <bracket value=")" linestart="7" colstart="10" lineend="7" colend="10" />

 </args\_list>

 </formal\_args\_list\_with\_or\_without\_const>

 <function\_body linestart="8" colstart="1" lineend="15" colend="1">

 <bracket value="{" linestart="8" colstart="1" lineend="8" colend="1" />

 <stmt\_list linestart="9" colstart="2" lineend="14" colend="10">

 <stmt\_list linestart="9" colstart="2" lineend="12" colend="16">

 <stmt\_list linestart="9" colstart="2" lineend="10" colend="11">

 <stmt\_list linestart="9" colstart="2" lineend="9" colend="11">

 <stmt\_or\_defined\_macro linestart="9" colstart="2" lineend="9" colend="11">

 <stmt linestart="9" colstart="2" lineend="9" colend="11">

 <variable\_declaration linestart="9" colstart="2" lineend="9" colend="10">

 <builtintype linestart="9" colstart="2" lineend="9" colend="4">

 <typename value="int" linestart="9" colstart="2" lineend="9" colend="4" />

 </builtintype>

 <assign linestart="9" colstart="6" lineend="9" colend="10">

 <identifier value="a" linestart="9" colstart="6" lineend="9" colend="6" />

 <operators value="=" linestart="9" colstart="8" lineend="9" colend="8" />

 <numeric value="1" linestart="9" colstart="10" lineend="9" colend="10" />

 </assign>

 </variable\_declaration>

 <semicolon value=";" linestart="9" colstart="11" lineend="9" colend="11" />

 </stmt>

 </stmt\_or\_defined\_macro>

 </stmt\_list>

 <stmt\_or\_defined\_macro linestart="10" colstart="2" lineend="10" colend="11">

 <stmt linestart="10" colstart="2" lineend="10" colend="11">

 <variable\_declaration linestart="10" colstart="2" lineend="10" colend="10">

 <builtintype linestart="10" colstart="2" lineend="10" colend="4">

 <typename value="int" linestart="10" colstart="2" lineend="10" colend="4" />

 </builtintype>

 <assign linestart="10" colstart="6" lineend="10" colend="10">

 <identifier value="b" linestart="10" colstart="6" lineend="10" colend="6" />

 <operators value="=" linestart="10" colstart="8" lineend="10" colend="8" />

 <numeric value="2" linestart="10" colstart="10" lineend="10" colend="10" />

 </assign>

 </variable\_declaration>

 <semicolon value=";" linestart="10" colstart="11" lineend="10" colend="11" />

 </stmt>

 </stmt\_or\_defined\_macro>

 </stmt\_list>

 <stmt\_or\_defined\_macro linestart="12" colstart="2" lineend="12" colend="16">

 <stmt linestart="12" colstart="2" lineend="12" colend="16">

 <expr\_function\_call linestart="12" colstart="2" lineend="12" colend="15">

 <identifier value="function" linestart="12" colstart="2" lineend="12" colend="9" />

 <bracket value="(" linestart="12" colstart="10" lineend="12" colend="10" />

 <expr\_list linestart="12" colstart="11" lineend="12" colend="14">

 <identifier value="a" linestart="12" colstart="11" lineend="12" colend="11" />

 <comma value="," linestart="12" colstart="12" lineend="12" colend="12" />

 <identifier value="b" linestart="12" colstart="14" lineend="12" colend="14" />

 </expr\_list>

 <bracket value=")" linestart="12" colstart="15" lineend="12" colend="15" />

 </expr\_function\_call>

 <semicolon value=";" linestart="12" colstart="16" lineend="12" colend="16" />

 </stmt>

 </stmt\_or\_defined\_macro>

 </stmt\_list>

 <stmt\_or\_defined\_macro linestart="14" colstart="2" lineend="14" colend="10">

 <stmt linestart="14" colstart="2" lineend="14" colend="10">

 <keyword value="return" linestart="14" colstart="2" lineend="14" colend="7" />

 <numeric value="0" linestart="14" colstart="9" lineend="14" colend="9" />

 <semicolon value=";" linestart="14" colstart="10" lineend="14" colend="10" />

 </stmt>

 </stmt\_or\_defined\_macro>

 </stmt\_list>

 <bracket value="}" linestart="15" colstart="1" lineend="15" colend="1" />

 </function\_body>

 </function\_definition\_without\_type>

 </stmt\_or\_defined\_macro>

 </stmt\_list>

</program>

# Приложение Г

Примеры ошибочных данных и сообщений, выдаваемых программой

Таблица 1. Ошибочные данные и сообщения, выдаваемые программой

| № | Текст ошибки (содержимое файла) | Описание ошибки | Выдаваемое пользователю сообщение |
| --- | --- | --- | --- |
| 1 | (txt-файл)5 019 5function | Ошибочные границы, находящиеся за пределами текста программы (текст программы см. приложение Б) | Wrong Border Input! Lower Border Is Outside The Program Code! |
| (txt-файл)15 019 5function | Wrong Border Input! Upper and Lower Borders Are Outside The Program Code! |
| 2 | (txt-файл)5 0 9 5 function | Некорректный формат файла: все данные вводятся одной строкой | Incorrect Format File! Data Are Entered In One Line! |
| (txt-файл)5 0 9 5 function | Некорректный формат файла: нарушена структура обеих границ | Incorrect Format File! Structure Of Both Borders Is Broken! |
| (txt-файл)5 0 9 5 function | Некорректный формат файла: нарушена структура одной границ | Incorrect Format File! Lower Border Structure Is Broken! |
| 3 | (txt-файл)5 09 5123func | Недопустимое имя для новой выделяемой функции | Invalid Function Name! |
| 4 | (txt-файл)5 89 5function | Невозможно провести рефакторинг из-за нарушения структуры участка кода (разрыв структуры if-else) | Impossible For Refactoring! Upper Border Breaks Statement If-Else! |
| 5 | (xml-файл)1. Отсутствует тег <program> </program> | Нарушение структуры дерева | Program Has Syntax Error! |
| 2. Главный модуль имеет идентификатор, отличный от main | Program Has No Main-Module! |
| 3. Возвращаемое значение main не соответствует значению оператора return | Wrong Return Type! |
| 4. Использование недопустимых тегов в дереве текста программы (см. ниже, под таблицей) | Использование многомерных массивов | Program Code Contains Unsupported Multi-Dimensional Array Data At Line <linestart>! |
| Использование оператора sizeof | Program Code Contains Unsupported Operator «Sizeof» At Line <linestart>! |
| Использование макросов | Program Code Contains Unsupported Macro Definition At Line <linestart>! |
| Использование шаблонов | Program Code Contains Unsupported Template Definition At Line <linestart>! |
| Использование using-директив | Program Code Contains Unsupported Using Namespace Definition At Line <linestart>! |
| Использование оператора множественного выбора switch | Program Code Contains Unsupported Operator «Switch-Case» At Line <linestart>! |
| Использование недопустимых квалификаторов типа | Program Code Contains Unsupported Type Qualifier At Line <linestart>! |
| Использование объединений | Program Code Contains Unsupported Custom Data Type – Union – At Line <linestart>! |
| Использование перечислений | Program Code Contains Unsupported Custom Data Type – Enum – At Line <linestart>! |
| Использование классов | Program Code Contains Unsupported Class Data Type At Line <linestart>! |
| Использование недопустимых типов данных | Program Code Contains Unsupported Data Type At Line <linestart>! |

Недопустимые данные и операции, и их теги и атрибуты в дереве текста программы:

1. использование многомерных массивов – описывается тегом <variable\_declaration> </variable\_declaration> без атрибутов, имеет внутри себя вложенный тег <lvalue> </lvalue>, внутри которого не допускается вложенность такого же одноименного тега;
2. использование оператора sizeof – описывается тегом <sizeof> </sizeof> без атрибутов, который имеет внутри себя другие вложенные теги;
3. использование макросов – описывается тегом <preprocessor … /> с атрибутом value с недопустимым значением "#define …";
4. использование шаблонов (template-производные) – описывается тегом <identified\_type\_meta\_specifier\_with\_template\_def> </identified\_type\_meta\_specifier\_with\_template\_def> без атрибутов, который имеет внутри себя вложенный тег <type\_meta\_specifier\_with\_template\_def> </type\_meta\_specifier\_with\_template\_def> без атрибутов;
5. использование using-директив (описание пространства имен) – описывается тегами <identifier …/> с атрибутом value равным "using" и <namespace\_definition\_start> </namespace\_definition\_start> без атрибутов, который имеет внутри себя другие вложенные теги, в частности тег <keyword…/> с атрибутом value равным "namespace";
6. использование оператора множественного выбора switch – описывается тегом <switch\_stmt> </switch\_stmt> без атрибутов, который имеет внутри себя другие вложенные теги;
7. использование недопустимых квалификаторов типа – описывается тегом <type\_with\_qualifier> </type\_with\_qualifier> без атрибутов, внутри которого есть вложенный тег <keyword …/> с атрибутом value равным "volatile";
8. использование объединений (union) – описывается тегом <identified\_type\_meta\_specifier\_with\_template\_def> </identified\_type\_meta\_specifier\_with\_template\_def> без атрибутов, внутри которого есть вложенный тег <keyword …/> с атрибутом value равным "union";
9. использование перечислений (enum) – описывается тегом <enum\_definition> </enum\_definition> без атрибутов, который имеет внутри себя другие вложенные теги;
10. использование классов, как часть объектно-ориентированного программирования – описывается тегом <identified\_type\_meta\_specifier\_with\_template\_def> </identified\_type\_meta\_specifier\_with\_template\_def> без атрибутов, внутри которого есть вложенный тег <keyword …/> с атрибутом value равным "class";
11. использование недопустимых типов данных – описывается тегом <builtintype> </builtintype> без атрибутов, который имеет внутри себя вложенный тег <typename …/> с атрибутом value с недопустимыми значениями "short", "unsigned", "signed", "long";

Таблица 2. Значения атрибутов тегов дерева

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Тег | Атрибут(ы) | Принимаемое значение | Сообщение об ошибке |
| 1 | [Любой] | linestart | Любая цифра ("0", "5" и т.д.). Любое другое значение влечет ошибку | Attribute «linestart» Of Tag <Tag> Has Invalid Value! |
| colstart | Attribute «colstart» Of Tag <Tag> Has Invalid Value! |
| lineend | Attribute «lineend» Of Tag <Tag> Has Invalid Value! |
| colend | Attribute «colend» Of Tag <Tag> Has Invalid Value! |
| 2 | <typename …/> | value | см. Недопустимые данные и операции, и их теги и атрибуты в дереве текста программы (пункт 11) |
| 3 | <keyword …/> | value | "const".Любое другое значение влечет ошибку | Attribute «value» Of Tag «keyword» Has Unsupported Value (xml-file, Line №№)! |
| 4 | <operators…/> | value | Любой оператор, допустимый в языке Си. Неподдерживаемый языком Си оператор влечет ошибку | Attribute «value» Of Tag «operators » Has Invalid Value (xml-file, Line №№)! |
| 5 | <identifier…/> | value | Идентификатор любых данных языка Си. Некорректное значение идентификатора влечет ошибку | Attribute «value» Of Tag «identifier » Has Invalid Value (xml-file, Line №№)! |
| 6 | <bracket…/> | value | "(", ")", "{", "}", "[", "]".Любое другое значение влечет ошибку | Attribute «value» Of Tag «bracket» Has Invalid Value (xml-file, Line №№)! |
| 7 | <comma…/> | value | ",".Любое другое значение влечет ошибку | Attribute «value» Of Tag «comma» Has Invalid Value (xml-file, Line №№)! |
| 8 | <semicolon…/> | value | ";".Любое другое значение влечет ошибку | Attribute «value» Of Tag «semicolon» Has Invalid Value (xml-file, Line №№)! |
| 9 | <numeric …/> | value | Любая цифра ("0", "5" и т.д.). Любое другое значение влечет ошибку | Attribute «value» Of Tag «numeric» Has Invalid Value (xml-file, Line №№)! |
| 10 | <character …/> | value | Любой символ. Другое значение влечет ошибку | Attribute «value» Of Tag «character» Has Invalid Value (xml-file, Line №№)! |
| 11 | <string …/> | value | "\"[набор\_символов]\"".Любое другое значение влечет ошибку | Attribute «value» Of Tag «string» Has Invalid Value (xml-file, Line №№)! |

Таблица 3. Вложенности составных тегов дерева

|  |  |  |  |
| --- | --- | --- | --- |
| № | Тег | Допустимые вложенные теги | Сообщение об ошибке |
| 1 | <builtintype> </builtintype> | <typename…/>.Наличие других вложенных тегов влечет ошибку | Tag «builtintype» Has Invalid Enclosure (xml-file, Line №№)! |
| 2 | <variable\_declaration> </variable\_declaration> | <builtintype> </builtintype>, <identifier…/> (возможны теги <lvalue> </lvalue>, <assign> </assign>).Наличие других вложенных тегов влечет ошибку | Tag «variable\_declaration» Has Invalid Enclosure (xml-file, Line №№)! |
| 3 | <lvalue> </lvalue> | <bracket value="["/>, <numeric…/>,<bracket value="]"/> (важен порядок!).Наличие других вложенных тегов влечет ошибку | Tag «lvalue» Has Invalid Enclosure (xml-file, Line №№)! |
| 4 | <assign> </assign> | <identifier …/>, <operators value="="/>,<identifier …/> (или <numeric…/>, <character…/>, <string …/>).Наличие других вложенных тегов влечет ошибку | Tag «assign» Has Invalid Enclosure (xml-file, Line №№)! |
| 5 | [Выражения](<expr\_lesser> </expr\_lesser>, <expr\_postfix\_increment> </expr\_postfix\_increment>,<expr\_function\_call> </expr\_function\_call> и т.д.) | Для математических и логических выражений:<identifier …/>, <operators value="="/>,<identifier …/> (или <numeric…/>, <character…/>, <string …/>).Для вызова функции:<identifier …/>, <bracket value="("/>,<expr\_list> </expr\_list><bracket value=")"/>.Наличие других вложенных тегов влечет ошибку | Tag «expr» Has Invalid Enclosure (xml-file, Line №№)! |
| 6 | <if\_then\_else> </if\_then\_else> | <if\_then> </if\_then> …/> (возможны теги <keyword value="else"/>, <stmt> </stmt>). Наличие других вложенных тегов влечет ошибку | Tag «if\_then\_else» Has Invalid Enclosure (xml-file, Line №№)! |
| 7 | <if\_then> </if\_then> | <keyword value="if"/>,<bracket value="("/>, [Выражение], <bracket value=")"/>, <stmt> </stmt>. Наличие других вложенных тегов влечет ошибку | Tag «if\_then» Has Invalid Enclosure (xml-file, Line №№)! |
| 8 | <for> </for> | <keyword value="for"/>,<bracket value="("/>, <variable\_declaration> </variable\_declaration>,<semicolon value=";"/>, [Выражение],<semicolon value=";"/>, [Выражение],<bracket value=")"/>, <stmt> </stmt>. Наличие других вложенных тегов влечет ошибку | Tag «for» Has Invalid Enclosure (xml-file, Line №№)! |
| 9 | <while> </while> | <keyword value="while"/>,<bracket value="("/>, [Выражение],<bracket value=")"/>, <stmt> </stmt>. Наличие других вложенных тегов влечет ошибку | Tag «while» Has Invalid Enclosure (xml-file, Line №№)! |
| 10 | <do\_while> </do\_while> | <keyword value="do"/>,<stmt> </stmt>,<keyword value="while"/>,<bracket value="("/>, [Выражение],<bracket value=")"/>, <semicolon value=";"/>. Наличие других вложенных тегов влечет ошибку | Tag «do\_while» Has Invalid Enclosure (xml-file, Line №№)! |
| 11 | <stmt\_list> </stmt\_list> | <stmt> </stmt>, а также теги, описанные выше (№№ 6-10) | Tag «stmt\_list» Has Invalid Enclosure (xml-file, Line №№)! |
| 12 | <stmt> </stmt> | Включает различные операторы языка Си, которые могут быть описаны через ключевые слова, например:<keyword value="return"/>,<identifier …/> (или <numeric…/>,<character…/>), <semicolon value=";"/>, или составной оператор, заключенный в фигурные скобки, например: <bracket value="{"/>, <stmt\_list> </stmt\_list><bracket value="}"/>, или любые выражения, типа:[Выражение],<semicolon value=";"/>. | Tag «stmt» Has Invalid Enclosure (xml-file, Line №№)! |
| 13 | [Объявление функции] | [Описание тегов, задающих объявление функции, и их атрибутов находится в приложении Д] | Tag <Tag> Has Invalid Enclosure (xml-file, Line №№)! |

# Приложение Д

Тэги и атрибуты XML-файла выделенной функции

Таблица 4. Тэги и атрибуты XML-файла выделенной функции

| № | Название | Тег | Атрибуты тега | Описание |
| --- | --- | --- | --- | --- |
| 1 | Блок операторов | <stmt\_list> </stmt\_list> | linestart, colstart, lineend, colend | Описание составного блока любых операторов. Атрибуты linestart, colstart обозначают «координату» начала выделяемого фрагмента; атрибуты lineend, colend обозначают «координату» конца выделяемого фрагмента (атрибуты linestart, colstart, lineend, colend присущи всем тегам). Имеет внутри себя другие вложенные теги (нижеописанные) |
| 2 | Определение функции без типа | <function\_definition\_ without\_type> </function\_definition\_ without\_type> | linestart, colstart, lineend, colend | Описание определения функции (заголовок и реализация) без типа. Имеет внутри себя другие вложенные теги (нижеописанные) |
| 3 | Реализация функции | <function\_body> </function\_body> | linestart, colstart, lineend, colend | Содержит описание функции. Имеет внутри себя другие вложенные теги (нижеописанные) |
| 4 | Формальный список аргументов | <formal\_args\_list> </formal\_args\_list> | linestart, colstart, lineend, colend | Имеет внутри себя вложенный тег <arg\_list> </arg\_list> |
| 5 | Список параметров | <arg\_list> </arg\_list> | linestart, colstart, lineend, colend | Описывает состав аргументов функции. Имеет внутри себя другие вложенные теги: <arg> </arg> |
| 6 | Описание одного параметра | <arg> </arg> | linestart, colstart, lineend, colend | Описывает один аргумент функции. Имеет внутри себя вложенные теги: <builtintype> </builtintype> (или <type\_with\_ptr> </type\_with\_ptr>), <identifier…/>. |
| 7 | Составной тип параметра с учетом указателя | <type\_with\_ptr> </type\_with\_ptr > | linestart, colstart, lineend, colend | Описывает составной аргумент, передаваемый по указателю. Имеет внутри себя вложенные теги: <type> </type>, <operators…/>, |
| 8 | Составной тип параметра | <type> </type> | linestart, colstart, lineend, colend | Описывает тип составного аргумента. Имеет внутри себя вложенные теги: <keyword…/>, <builtintype> </builtintype>. |
| 9 | Построение типа параметра  | <builtintype> </builtintype> | linestart, colstart, lineend, colend | Имеет внутри себя вложенный тег <typename…/> |
| 10 | Простой тип параметра | <typename…/> | value, linestart, colstart, lineend, colend | Описывает тип параметра. Атрибут принимает значения только "bool", "int", "float", "double", "char". Вложенных тегов не имеет |
| 11 | Ключевое слово | <keyword…/> | value, linestart, colstart, lineend, colend | Описывает ключевые слова языка Си. Атрибут принимает значения ключевых слов. Вложенных тегов не имеет. |
| 12 | Оператор | <operators…/> | value, linestart, colstart, lineend, colend | Описывает операцию над данными. Вложенных тегов не имеет |
| 13 | Присвоение | <assign> </assign> | linestart, colstart, lineend, colend | Операция присвоения. Имеет внутри себя другие вложенные теги |
| 14 | Идентификатор параметра/ функции | <identifier…/> | value, linestart, colstart, lineend, colend | Описывает идентификатор данных (имя функции/переменной и т.д.). Вложенных тегов не имеет |
| 15 | Скобка | <bracket…/> | value, linestart, colstart, lineend, colend | Описывает любой вид скобок (открывающая/ закрывающая, круглая/ квадратная/фигурная). Вложенных тегов не имеет |
| 16 | Запятая | <comma…/> | value, linestart, colstart, lineend, colend | Описывает запятую. Атрибут принимает только значение ",".Вложенных тегов не имеет |
| 17 | Точка с запятой | <semicolon …/> | value, linestart, colstart, lineend, colend | Описывает точку с запятой. Атрибут принимает только значение ";".Вложенных тегов не имеет |
| 18 | Числовое значение | <numeric …/> | value, linestart, colstart, lineend, colend | Описывает числовую константу. Атрибут принимает значение любой цифры. Вложенных тегов не имеет |
| 19 | Символьное значение | <character …/> | value, linestart, colstart, lineend, colend | Описывает символьную константу. Атрибут принимает значение любого символа. Вложенных тегов не имеет |
| 20 | Описание переменной | <variable\_declaration> </variable\_declaration> | linestart, colstart, lineend, colend | Содержит описание переменной. Имеет внутри себя другие вложенные теги |
| 21 | Оператор | <stmt> </stmt> | linestart, colstart, lineend, colend | Содержит описание оператора (декларация и необходимые параметры). Имеет внутри себя другие вложенные теги |
| 22 | Вызов функции | <expr\_function\_call> </expr\_function\_call> | linestart, colstart, lineend, colend | Описание вызова функции с необходимыми параметрами. Имеет внутри себя другие вложенные теги |
| 23 | Аргументы, передаваемые в функцию | <expr\_list> </expr\_list> | linestart, colstart, lineend, colend | Описание аргументов, передаваемых в функцию. Имеет внутри себя другие вложенные теги |
| 24 | Аргумент функции, передающийся по указателю | <expr\_dereference> </expr\_dereference> | linestart, colstart, lineend, colend | Описание одного составного аргумента, передаваемого в функцию по указателю. Имеет внутри себя вложенные теги <operators…/>, <identifier…/> |

Пример XML-дерева заголовка функции:

bool func(int a, const char b, double \*c)

{ … }

<stmt\_list>

 <builtintype>

 <typename value="bool" />

</builtintype>

<function\_definition\_without\_type>

 <identifier value="function" />

 <formal\_args\_list>

 <bracket value="(" />

 <arg\_list>

 <arg>

 <builtintype>

 <typename value="int" />

 </builtintype>

 <identifier value="a" />

 </arg>

 <comma value="," />

 <arg>

 <type>

 <keyword value="const" />

 <builtintype>

 <typename value="int" />

 </builtintype>

 </type>

 <identifier value="b" />

 </arg>

 <comma value="," />

 <arg>

 <type\_with\_ptr>

 <builtintype>

 <typename value="double" />

</builtintype>

 <operators value="\*" />

 </type\_with\_ptr>

 <identifier value="b" />

 </arg>

 </arg\_list>

 <bracket value=")" />

 </formal\_args\_list >

<function\_body>

 <bracket value="{" />

 <stmt\_list>

 …

 …

 …

 </stmt\_list>

 <bracket value="}" />

</function\_body>

</function\_definition\_without\_type>

</stmt\_list>

Пример XML-вызова данной функции:

function(var1, \*var2, var3);

<stmt>

<expr\_function\_call>

<identifier value="function" />

<bracket value="(" />

<expr\_list>

 <identifier value="var1" />

<comma value="," />

<expr\_dereference>

 <operators value="\*" />

 <identifier value="var2" />

</expr\_dereference>

<bracket value="," />

 <identifier value="var3" />

</expr\_list>

<bracket value=")" />

</expr\_function\_call>

<semicolon value=";" />

</stmt>